CSE 332

Data Structures and Parallelism

P vs. NP: Efficient Reductions Between Problems

More Graph Problems

Let's consider the **longest path** problem on a graph.

Remember, we were able to do **shortest paths** using Dijkstra's.

Take a few minutes to try to solve the **longest path** problem.

Decision Problems

Definition (Decision Problem)

A **decision problem** (or **language**) is a set of strings ($L \subseteq \Sigma^*$).
An algorithm (from Σ^* to `boolean`) solves a decision problem when it outputs `true` iff the input is in the set.

PRIMES	
Input(s):	Number x
Output:	`true` iff x is prime

An Algorithm that solves PRIMES

```
1 isPrime(x) {
2    for (i = 2; i < x; i++) {
3       if (x % i == 0) {
4          return false;
5       }
6    }
7    return true;
8 }
```

Efficient?

In this lecture, we'll be talking about **efficient reductions**. So, naturally, we have to answer two questions:

- What is an efficient algorithm?
- What is a reduction?

Efficient Algorithm

We say an algorithm is **efficient** if the worst-case analysis is a **polynomial**. Okay, but...

- $n^{10000000\ldots}$ is polynomial
- $3000000000000000n^3$ is polynomial

Are those really efficient?
Well, no, but, in practice...

> when a polynomial algorithm is found the constants are actually low

Polynomial runtime is a **very** low bar, if we can't even get that...

Reductions

This lecture is about exposing **hidden** similarities between problems.

We will show that problems that are **cosmetically different** are **substantially the same**!

Our main tool to do this is called a **reduction**:

Reductions

We have two **decision problems**, **A** and **B**. To show that **A** is "at least as hard as" **B**, we

- Suppose we can solve **A**
- Create an algorithm, which calls **A** as a method, to solve **B**

To show they're the same, we have to do both directions.

Longest Paths and HAM! 5

Two New Computational Problems

LONG-PATH
Input(s): Unweighted Graph G; Number k
Output: `true` iff G has a path with k edges

HAM-PATH
Input(s): Unweighted Graph G
Output: `true` iff G has a path using all vertices

Suppose we could solve **LONG-PATH**...

"Algorithm"

```
1 HAM-PATH(G) {
2    return LONG-PATH(G, |V| - 1)
3 }
```

Suppose we could solve **HAM-PATH**...

"Algorithm"

```
1 LONG-PATH(G, k) {
2    for (G' =(v_1,v_2,...,v_k) in G) {
3       if (HAM-PATH(G')) {
4          return true;
5       }
6    }
7    return false;
8 }
```

A 2-CRAYOLA Question 6

Definition (k-coloring)

A k-**coloring** of a graph G is an assignment of k colors to vertices such that no two adjacent vertices have the same color.

2-COLOR
Input(s): Graph G
Output: `true` iff G has a valid 2-coloring

Can we solve this?

Algorithm For 2-COLOR

Try all 2^n possible colorings of the input graph!

Can we solve this efficiently?

Efficient Algorithm For 2-COLOR

Do a `dfs` on the graph! Every time we hit a vertex, assign it the opposite color from the vertex we just visited. If there's a color conflict, output `false`. If we finish with no color conflict, output `true`.

A 3-CRAYOLA Question 7

Definition (k-coloring)

A k-**coloring** of a graph G is an assignment of k colors to vertices such that no two adjacent vertices have the same color.

3-COLOR
Input(s): Graph G
Output: `true` iff G has a valid 3-coloring

Inefficient Algorithm For 3-COLOR

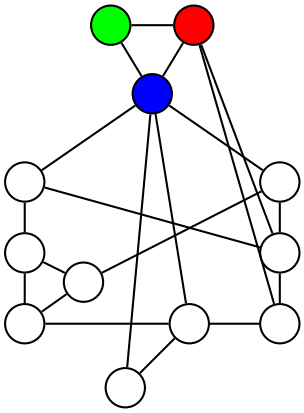
Try all 3^n possible colorings of the input graph!

Efficient Algorithm For 3-COLOR

UNKNOWN

A Graph Called "Gadget" 8

Find a valid 3-coloring of this graph. To orient ourselves, I've started it:



Another Decision Problem! 9

CIRCUITSAT
Input(s): n-Input/1-Output Circuit C
Output: `true` iff C has a satisfying assignment

Inefficient Algorithm For CIRCUITSAT

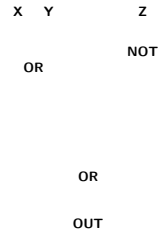
Try all 2^n possible assignments of variables

Efficient Algorithm For CIRCUITSAT

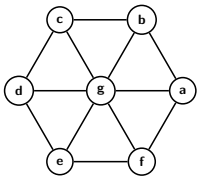
UNKNOWN

Suspicious... 10

CIRCUITSAT



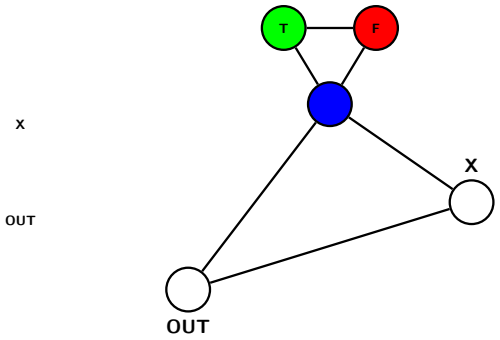
3-COLOR



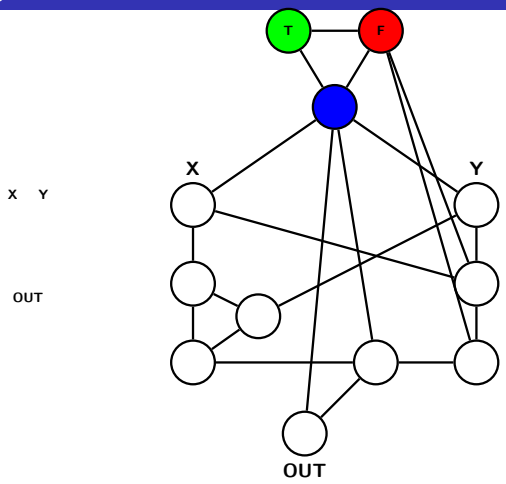
We don't know how to solve either of these problems...

Could they be the same problem in disguise?

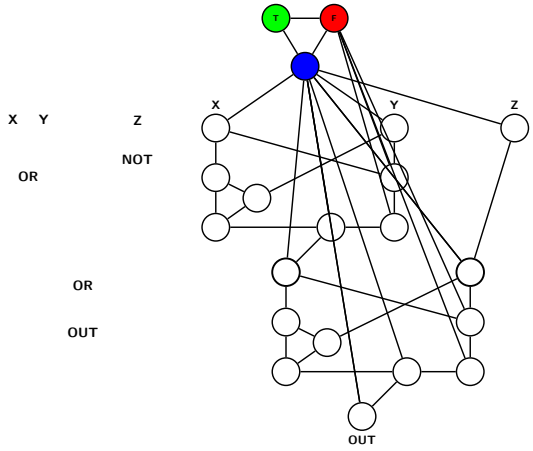
Not Gadget with Labels



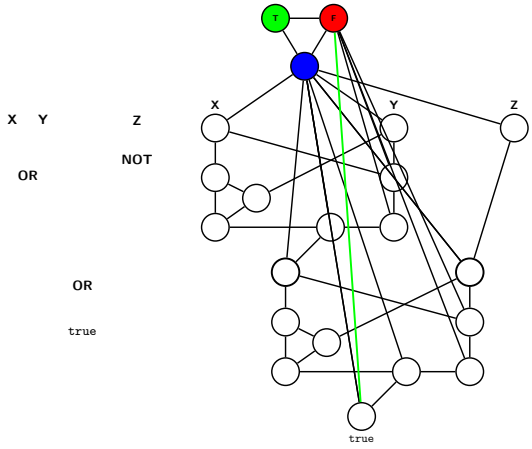
Or Gadget with Labels



Circuit



SATISFIABLE Circuit



Lesson

We found a way to "emulate" circuit satisfiability using three coloring!

If we can find a solution to **3-COLOR**, we can solve **CIRCUITSAT** quickly.

These problems are substantially the same